Assignment 3 , 159.735, 2020 S2

Family Name: JIANG , Given Name: TAO ,

Student ID:20004769

**Assignment3 :** **Parallel Solution of the Heat Distribution Problem**

The experiment Develop an OpenMP parallel solution the distribution of temperatures across a printed circuit plate.

OpenMP can realize parallel computing with minimal changes to the original program. It dramatically reduces the cost of secondary development. If exceeding the limit of the number of virtual threads, and all threads need to wait in line. Only when there are free threads will the execution continue. It increases the total running time.

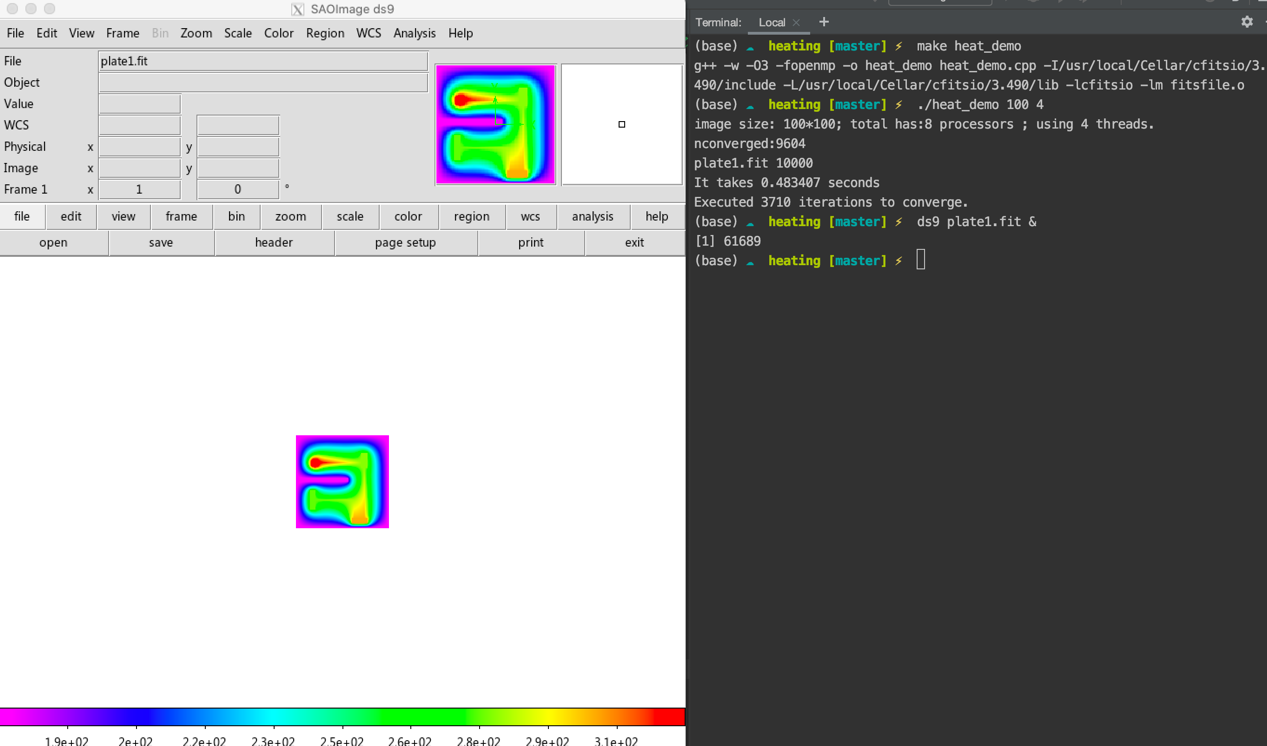
* **User’s Guide**

I use Ubuntu as a development platform. Furthermore, it only supports 12 processors maximum. The fist param means image size, and the second param is the number of threads.

Ubuntu env：

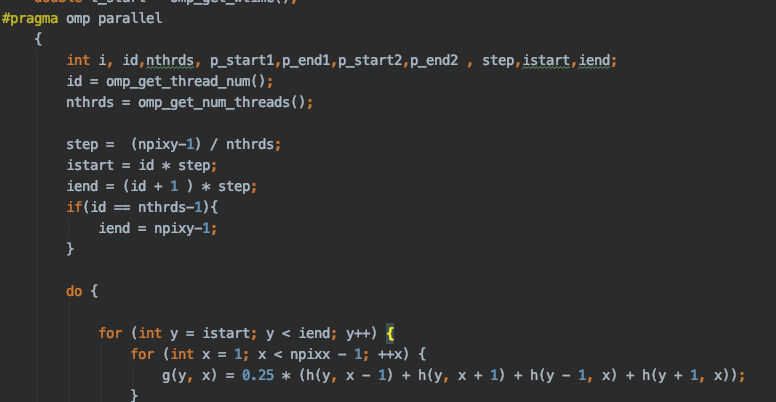
```shell  
make heat\_demo  
./heat\_demo 100 4   
ds9 plate11.fit &  
```

There is a screenshot of the metal plate's heat distribution after running the solution by 4 threads in an image size of 100\*100.

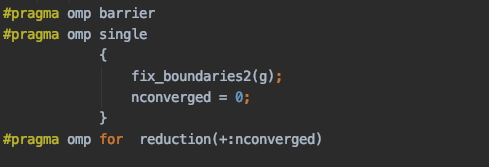


* **Experiment results**

The program of this experiment uses partition to achieve parallel computing. Ensure parallel orderliness.



Use *#pragma omp barrier* to wait for the end of execution of all threads, and use one thread to continue execution. Through *#pragma omp for reduction(+:nconverged)*, make the *nconverged* can be shared among threads.



The experiment uses a 3-group, controlled experiment. Using the image sizes of 100\*100, 200\*200 and 400\*400 to test. The experimental environment is the Ubuntu system with 12processors. The time required for convergence under different Threads recorded in the experiment.

The results are as follows:

|  |  |  |  |
| --- | --- | --- | --- |
| **Image size** | **Seq Time** | **Threads** | **parll Time** |
| 100\*100 | 0.0642768 | 1 | 0.0784799 |
| 100\*100 | 0.0664076 | 2 | 0.058229 |
| 100\*100 | 0.067696 | 3 | 0.0521353 |
| 100\*100 | 0.0679974 | 4 | 0.0508455 |
| 100\*100 | 0.066589 | 5 | 0.0506563 |
| 100\*100 | 0.0642009 | 6 | 0.0538813 |
| 100\*100 | 0.0685763 | 7 | 0.0547991 |
| 100\*100 | 0.0661796 | 8 | 0.0565203 |
| 100\*100 | 0.0651221 | 9 | 0.0543733 |
| 100\*100 | 0.06936; | 10 | 0.058209 |
| 100\*100 | 0.0693396 | 11 | 0.0571379 |
| 100\*100 | 0.0681096 | 12 | 0.10362 |
| 100\*100 | 0.0679177 | 13 | 0.408107 |
| 100\*100 | 0.0653923 | 14 | 0.432317 |
| 100\*100 | 0.0686017 | 15 | 0.46674 |
| 100\*100 | 0.0685092 | 16 | 0.491517 |
| 100\*100 | 0.0682336 | 17 | 0.521056 |
| 100\*100 | 0.0712754 | 18 | 0.55393 |
| 100\*100 | 0.0641524 | 19 | 0.622513 |
| 100\*100 | 0.0685478 | 20 | 0.660634 |
| 100\*100 | 0.0678403 | 22 | 0.724647 |
| 100\*100 | 0.069513 | 24 | 0.778123 |
| 100\*100 | 0.0685193 | 32 | 1.0274 |
| 100\*100 | 0.0687846 | 36 | 1.14522 |
| 100\*100 | 0.0689726 | 42 | 1.32395 |
| 100\*100 | 0.0691622 | 48 | 1.48582 |
| 100\*100 | 0.0670827 | 52 | 1.62032 |
| 100\*100 | 0.0692181 | 60 | 1.85398 |

|  |  |  |  |
| --- | --- | --- | --- |
| **Image size** | **Seq Time** | **Threads** | **parll Time** |
| 200\*200 | 0.912005 | 1 | 0.904323 |
| 200\*200 | 0.772083 | 2 | 0.494656 |
| 200\*200 | 0.792958 | 3 | 0.428235 |
| 200\*200 | 0.779319 | 4 | 0.398006 |
| 200\*200 | 0.739738 | 5 | 0.38067 |
| 200\*200 | 0.848932 | 6 | 0.372188 |
| 200\*200 | 0.958478 | 7 | 0.448321 |
| 200\*200 | 0.959776 | 8 | 0.456724 |
| 200\*200 | 0.909432 | 9 | 0.436946 |
| 200\*200 | 0.938246 | 10 | 0.451774 |
| 200\*200 | 0.951516 | 11 | 0.414522 |
| 200\*200 | 0.90971 | 12 | 0.454938 |
| 200\*200 | 0.934964 | 13 | 1.77995 |
| 200\*200 | 0.932305 | 14 | 1.90045 |
| 200\*200 | 0.916378 | 15 | 1.97375 |
| 200\*200 | 0.869729 | 16 | 2.07069 |
| 200\*200 | 0.837667 | 17 | 2.16456 |
| 200\*200 | 0.818257 | 18 | 2.26736 |
| 200\*200 | 0.831007 | 19 | 2.37528 |
| 200\*200 | 0.772535 | 20 | 2.47244 |
| 200\*200 | 0.842878 | 22 | 2.70092 |
| 200\*200 | 0.801453 | 24 | 2.9328 |
| 200\*200 | 0.814322 | 32 | 3.77537 |
| 200\*200 | 0.813363 | 36 | 4.2083 |
| 200\*200 | 0.812357 | 42 | 4.82122 |
| 200\*200 | 0.820017 | 48 | 5.44054 |
| 200\*200 | 0.791133 | 52 | 6.07795 |
| 200\*200 | 0.936731 | 60 | 6.6507 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Image size** | **Seq Time** | **Threads** | **parll Time** | **speedup** |
| 400\*400 | 11.4228 | 1 | 9.62223 | 1.1871261 |
| 400\*400 | 10.0197 | 2 | 9.62968 | 1.0405019 |
| 400\*400 | 9.69053 | 3 | 5.9071 | 1.6404886 |
| 400\*400 | 9.83095 | 4 | 4.78611 | 2.0540585 |
| 400\*400 | 10.2262 | 5 | 4.10372 | 2.4919341 |
| 400\*400 | 9.81344 | 6 | 3.72352 | 2.6355277 |
| 400\*400 | 9.81124 | 7 | 3.48797 | 2.8128797 |
| 400\*400 | 9.8035 | 8 | 4.35389 | 2.2516646 |
| 400\*400 | 11.0676 | 9 | 4.2359 | 2.6128096 |
| 400\*400 | 9.97423 | 10 | 3.99004 | 2.499782 |
| 400\*400 | 9.96672 | 11 | 3.94217 | 2.5282319 |
| 400\*400 | 9.78349 | 12 | 3.73047 | 2.6225891 |
| 400\*400 | 10.0734 | 13 | 3.69528 | 2.7260181 |
| 400\*400 | 10.4151 | 14 | 11.4354 | 0.9107771 |
| 400\*400 | 9.74194 | 15 | 10.8997 | 0.8937806 |
| 400\*400 | 9.76538 | 16 | 10.7661 | 0.907049 |
| 400\*400 | 11.4897 | 17 | 10.584 | 1.0855726 |
| 400\*400 | 9.84455 | 18 | 10.7459 | 0.9161215 |
| 400\*400 | 9.85956 | 19 | 10.932 | 0.901899 |
| 400\*400 | 9.79357 | 20 | 11.1345 | 0.8795698 |
| 400\*400 | 9.68574 | 22 | 11.4982 | 0.8423701 |
| 400\*400 | 9.70191 | 24 | 11.711 | 0.8284442 |
| 400\*400 | 9.73055 | 32 | 12.1167 | 0.8030693 |
| 400\*400 | 9.97385 | 36 | 14.1013 | 0.7073 |
| 400\*400 | 10.0007 | 42 | 15.1929 | 0.6582483 |
| 400\*400 | 9.92384 | 48 | 19.3752 | 0.5121929 |
| 400\*400 | 9.96688 | 52 | 21.665 | 0.4600452 |
| 400\*400 | 10.0962 | 60 | 23.8182 | 0.4238859 |

Table 1-3: The test result of 100\*100, 200\*200 and 400\*400 image size

Chart 1: 100\*100 image size time cost

Chart 2: 200\*200 image size time cost

Chart 3: 400\*400 image size time cost

Chart 4: 400\*400 image size speedup

* **Conclusion**

It can be seen from Chart 1 that the performance of a parallel program with 2-12 threads is almost the same as the time of sequential. The reason is the image size is too small, the program execution speed is fast, and the speed of parallel computing is not apparent. However, it can be seen in Chart 2 and Chart 3 that the running time of the 2-12threads parallel program is less than the sequential running time. Within the scope of virtual processors, parallel programs can be speeded up well.

When using more than 12 threads in parallel computing, the results of the three groups of data all show the same trend that is the parallel computing time is higher than the sequential computing time. The reason is after exceeding the limit of the number of virtual threads, and all threads need to wait in line. Only when there are free threads will the execution continue. It will increase the total running time.

Figure 4 is the speed-up ratio of 400\*400 parallel programs. In the running results of 1-6 threads, the speed-up trend is declining. The greater the number of threads, the less noticeable the speed-up effect. It is in line with Amdahl's Law. Due to runtime errors, the running trend of 6-12threads does not conform to the conclusion of Amdahl's Law. When the program execution exceeds 12threads, the threads need to be queued for execution. The experimental results do not conform to the conclusion of Gustafson's Law. Gustafson's Law's conclusion is only valid within the range of the number of virtual threads.